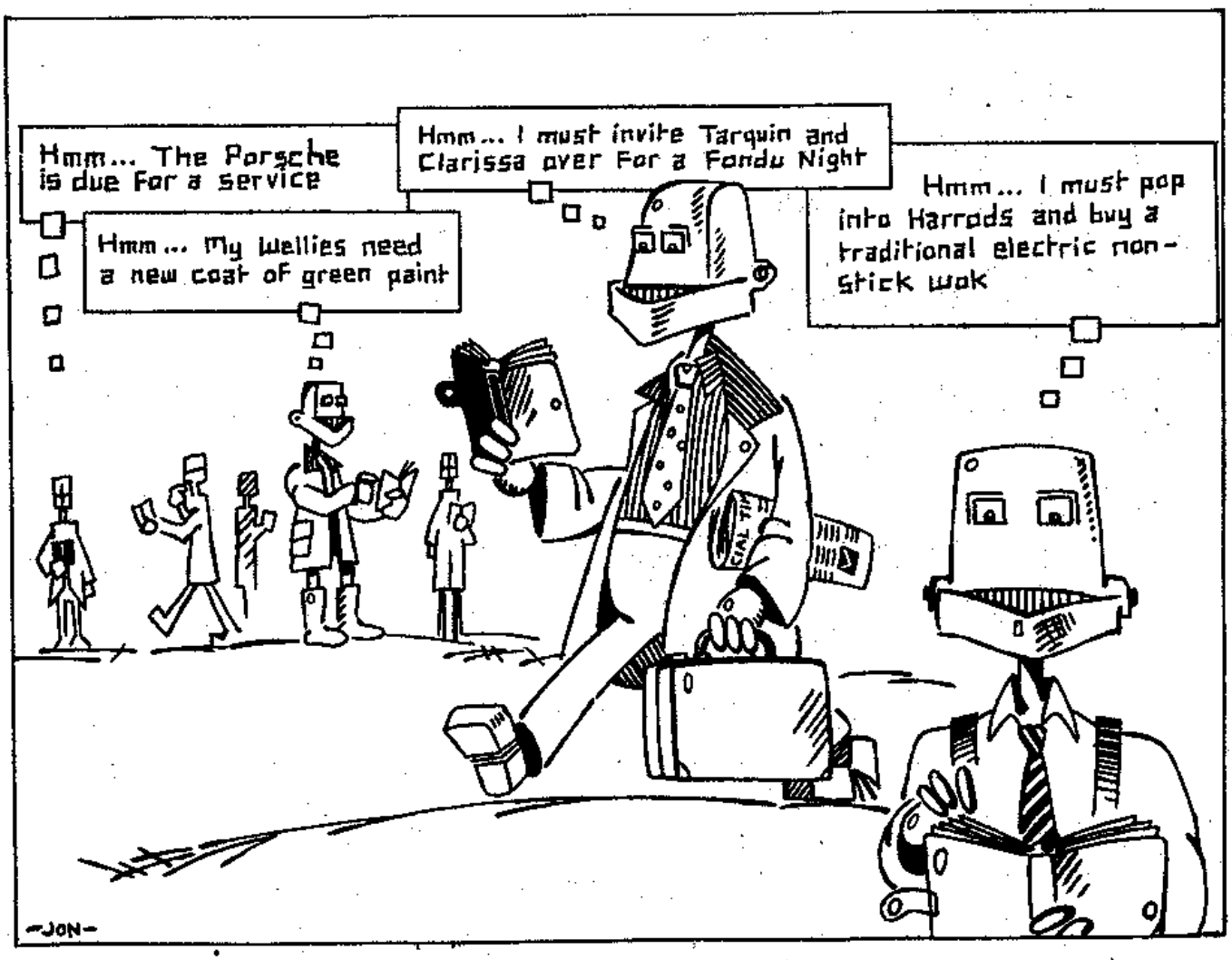
Vol 2 - No 8 April 1989.

FORMAT

THE MONTHLY MAGAZINE FOR SPECTRUM, DISCiPLE & PLUS D USERS



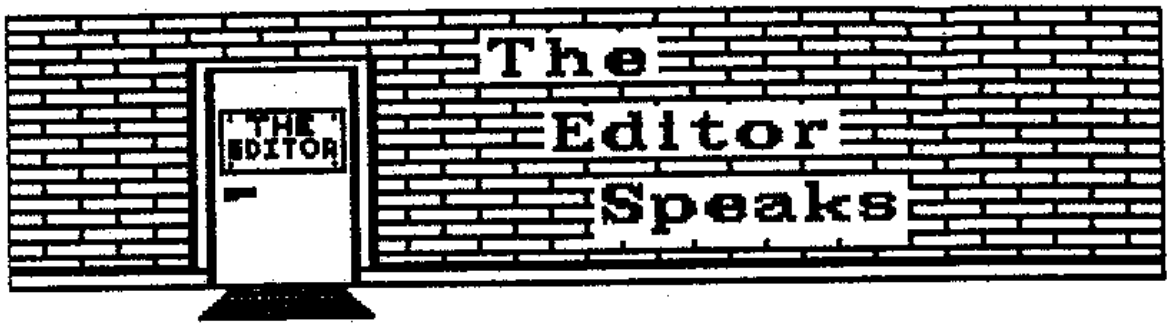
FILOFAX PAGES From Your Spectrum

Vol 2 No 8. CONTENTS Apr 1989.

THIS MONTHS ADVERTISERS:-

FORMAT is published by INDUG. 34 Bourton Road, Gloucester, GL4 0LE, England. Telephone 0452-412572. **DISCiPLE** and **PLUS D** are trade marks of MILES GORDON TECHNOLOGY, Lakeside Technology Park, Phoenix Way, Swansea, South Wales, SA7 9EH. Telephone 0792-791100. The DISCiPLE is marketed by Rockfort Products, 81 Church Road, Hendon, London, NW4 4DP.

Printed by D.S.LITHO. Gloucester. Tel:- (0452) 23198



FORMAT starts to get a new look this month. Pages are sprouting columns just like this one. Why? I think it looks better, and in addition it enables me to get more on a page. By using a slightly smaller type face, printing at 12 characters to the inch instead of 10, more text can be printed on each line. But if this was done as one long line (80 characters) it would make FORMAT almost impossible to read. So, after a long period of experimenting, columns come to these pages. Some of this months magazine, which were pre-set before the new print wheels arrived, are still in single (wide) column, but from next month most (if not all) will have the new look. I would welcome readers comments on the change.

The observant amongst you will notice that this is the April issue. Where has the March issue gone? Did the post office eat it? Has someone changed the calendar this year and left out March? Well no, none of those, its my doing. Since January 1988 I have been trying to get back to publishing at the beginning of the month (as we had been up to then) but each month its proved impossible. Publishing around the 21st of the month means I only have one week where the issue bears the same month as the one I am working in. This leads to confusion (especially with new members) and has made life very difficult when it comes to selling advertising space. So FORMAT Vol 2 No 8 becomes April instead of March. Dont worry its only a cosmetic change, it makes no difference to your subscriptions, you still get the right number of issues.

Many people have asked the question 'What happened to The Help Page'. It has not appeared in FORMAT since last August simply because it was always written by yours truly and there just hasn't been time over the last few months to collate and organize the page. But now Nev Young has volunteered (well, after a bit of bullying anyway) to produce a Help Page each month. As some questions have been answered by articles over the months I think it is best to allow Nev to start with a clean sheet. So starting now I will pass on all letters requesting help to Nev. His first page should appear next month. If you wrote to me with a problem over a month ago it may be better to restate your question in a fresh letter to Nev Young. Letters will only be answered through the Help Page.

That also reminds me, to remind you, that I can't give personal replies to letters, no matter what they are about. If you need to contact me please use the HOTLINE, thats what its there for.

John Wase has also asked me to remind you that he URGENTLY needs small programming items for his new 'SHORT STOP' feature. The response from last months request was almost non-existent. So come on, any small items (Basic, Machine Code, anything), just send them to John at the address give last month.

Last item this month. Readership still continues to grow, we are fast approaching 1400, but could still grow even faster now that we have a wider Spectrum coverage. If you have any ideas on how we could attract more subscribers then please let me know. The larger the readership the faster FORMAT can grow. I might even manage to rustle up a reward for the best ideas.

See you next month.

Bob Brenchley. Editor.

NEWS ON 4

SUSIE GROUP OPEN MEETING.

SUSIE, the spectrum educational group plan a meeting in Birmingham in mid April. John Croghan, secretary of SUSIE will be organizing the meeting to bring together teachers, educationalists and parents who have an interest in using the Spectrum in education. Details of the meeting and SUSIE membership can be obtained from John Croghan, Head, St. Francis School, Teazel Avenue, Bournville, Birmingham.

Anyone who has already contacted SUSIE, following our news item in December, will be receiving details in the near future.

MGT LAUNCH UPGRADEABLE DRIVE.

Fed up with changing your disc drive every time you change your computer? Well MGT now have the answer for you. A disc drive that will work with most major systems.

Simply by changing the cable (each one available separately) you could change your drive from a BBC to an IBM, or an Atari ST.

Full details are available from MGT on 0792-791100.

MIRA PASCAL AND FORTRAN.

MIRA software have release versions of PASCAL and FORTRAN for the Spectrum. The Leicestershire based company sell the programs on tape for ease of transfer to both Microdrive and disc systems.

PASCAL is produced to the BS 6192 standard while FORTRAN appears to be somewhere between the FORTRAN IV and the FORTRAN 77 standards. Both have their own editors and come with small, but well written, manuals. They cost £15 each and are available from MIRA Software, 24 Home Close, Kibworth, Leics, LE8 0JT.

AMSTRAD SELL OUT.

No, I'm sorry to say Sugar has not done the honourable thing and sold his Sinclair computer operation to someone who knows how to build computers. Instead Database Publications has sold its three 'official' magazines: AMSTRAD COMPUTER USER, AMSTRAD PROFESSIONAL COMPUTING and AMSTRAD PCW MAGAZINE, to FOCUS Publications for an undisclosed sum. What plans Focus have are unclear as is the future of the OFFICIAL status.

Z88 PRICE CUT.

Sinclair has now officially cut the price of his excellent lap-top, the Z88, to £199.95

Comets and Dixons had reduced their prices after Christmas but the cut by Cambridge Computers is not linked. Instead Sinclair points towards increased sales and reduced building costs as being the main reason for the cuts.

SINCLAIR ON TV.

The BBC are planning to make a T.V. program on the ups and downs of Uncle Clive. Lets hope the program concentrates on how he, almost single handed, gave the world AFFORDABLE home computing.

JOB CUTS AT AMSTRAD

Eight staff are being made redundent from Amstrad's Brentwood H.Q. According to Amstrad this is to improve efficiency. Lots of Amstrad users would, I think, rather these people were transferred to customer support.

If you have any news items you want to pass on then send them in. Please mark the envelope NEWS in the top corner.

HACK-ZONE

By: Hugh J. McLenaghan.

This months article is on BASIC program protection. The following program is designed to load in programs which cannot be MERGED into memory.

```
 10 CLEAR 65535
 20 PRINT AT 10,11;"LOAD BASIC"
 30 RESTORE
 40 FOR N=23296 TO 23370: READ A: POKE N,A: NEXT N
 50 RANDOMIZE USR 23296
 60 DATA 237,91,83,92,42,89,92,43,205,229,25
 70 DATA 221,33,75,91,17,17,0,175,55,205,86,5
 80 DATA 56,2,207,26,221,33,75,91,221,126,0,254,0
 90 DATA 32,229,42,83,92,237,75,86,91,205,85,22
100 DATA 42,83,92,237,91,90,91,25,34,75,92,221,42
110 DATA 83,92,237,91,86,91,62,255,55,205,86,5,207,255,201
```

Save this as soon as you can (before you run it) in case of typing errors. What you do now is RUN it and start the tape of any protected program.

After the BASIC has loaded it will give the report **0 OK 0:1.** You can now save it off to disc and you should also be able to LIST it.

Now I will show you how to protect your own BASIC programs. The first method is a simple method of 'in system' protection. It is for 48K BASIC only. As a first line number in you program put:-

Line Number followed by the keypresses to get INK 7 and PAPER 7 in the line (see article by Clyde Bish - FORMAT Vol 2 No 2), then type **POKE 23570,2**

After the program has been run the protection has taken effect. If you now try to break into the program YOU CAN, but after you press any key you will get a continuous noise which continues until you turn the computer off.

Method 2. This is also 'in system'. It is a common method used in loaders. You use **POKE 23659,0** & **POKE 23613,0.** Both of these pokes cause the system to crash if you try to break into the program. What happens is this. POKE 23659,0 sets the number of lines in the lower screen to zero, the system crashes because it has no where to print the error message. Just in case the system resets this value the POKE 23613 overwrites the ERR SP pointer which is used to return to operating system after an error has occurred. These methods, of course, do not prevent you from MERGEing them in and removing these commands before they have chance to take effect.

What we need is a MERGE proof loader. This is very simple and also

has a few different methods.

Method 1. First enter the line:- **1 REM**

Then type in as a direct command:-

(DISCiPLE) **POKE @23755-664,65535**
(PLUS D) **POKE @23755-8192,65535**

The REM line will now vanish, but it is still there. All you need to do now is to type in your program and save it as normal. Now if you try to MERGE this program it will cause the computer to crash. This happens because the computer does not know how to cope with a line number greater than 32767!

Method 2. Type in your program as normal, when complete type in the following before SAVEing :-

(DISCiPLE) **POKE @23757-664,65535**
(PLUS D) **POKE @23757-8192,65535**

When you attemt to MERGE this program you will get the out of memory message because the first line now has a length of 64K.

Now for some fun. Type in the following program EXACTLY as listed.

```
  10 CLS
  20 PRINT "This is some fun!!"
  30 FOR A=1 TO 400
  40 GO SUB 16384
  50 POKE 23692,-1
  60 PRINT "Nice fun isn't it! ";
  70 NEXT A
  80 INK 0: PAPER 7: BORDER 7
  90 STOP
9997 LET A=PEEK 23637+256*PEEK 23638: POKE @A-664,16384: STOP
9998 INK (INT (RND*8)+1): PAPER (INT (RND*8)+1): RETURN
```

If using a PLUS D then change the 664 in line 9997 to read 8192. This is the base address for the POKE @ command.

Next type RUN 9997, you will get an OK message, then delete 9997. List the program and examine it carefully. You will notice that what was line number 9998 has now changed to a funny looking mess. This mess actually means 16384. RUN it to see what I mean.

If you wish to send tips, comments, ideas, conversions please send them to me directly at the following address. Please remember to include a stamped addressed envelope.

Hugh J. McLenaghan,
HACK ZONE,
36 Floorsburn Crescent,
Johnstone,
Renfrewshire,
PA5 8PF.

See you next month!!

HISOFT C

By: John Hamilton.

This is not a review of the **C** language. The arguments on the relative merits of PASCAL, Modula-2, **C** and other languages have been covered in many articles in other magazines and I do not intend to air my views here. This one looks at the **Hisoft C** compiler and how to use it on the DISCiPLE and PLUS D. Some of the points are covered in the documentation, but as some of them may be a little unclear, there is no harm in going over them again.

THE DISTRIBUTION DISC

You will probably get a disc with the following files on it:-

1	autoload	BAS	**you will need to change this**
2	cc.code	CDE	
3	cc.scr	SCREEN$	
4	hc.code	CDE	**you won't need this**
5	patch.code	CDE	
6	Sys 3d	CDE	**or this**
7	conv-2	BAS	**fixes for older PLUS D**
8	conv-1	BAS	**fixes for older DISCiPLE**
9	stdio.h	OPENTYPE	
10	stdio.lib	OPENTYPE	

SETTING UP YOUR SYSTEM

Before you start to use the compiler, you will need to get your system up-to-date. The following table tells you what to do:-

	Version	
	1	get your ROM replaced
	2	" " " "
DISCiPLE	3a	run **conv-1** on your system
	3b	" " "
	3c	" " "
	3d	do nothing - your system is correct
	1	delete line 399 and run **conv-2**
PLUS_D	1a	do nothing - your system is correct
	2	run **conv-2** on your system
	2a	do nothing - your system is correct

You should set up a runnable disc as follows.

1. Load your system normally (do not use the system on the distributed disc - it is not initialised for your configuration).

2. Format yourself a new disc

3. Write protect the distribution disc

4. If you need to, LOAD **conv-1** or **conv-2** from the distribution disc and, if necessary (check the comments at the start of the programs) modify the program for your configuration.

5. Load your newly-formatted disc into drive one and either

   RUN conv-1 or conv-2. (note that they save your new system code to D1 with the correct name when the updates have been done)

   or one of the following, if your system is..

   SAVE d1"SYS 3D"**CODE** 0,6656 DISCiPLE
   SAVE d1"+SYS 1A"**CODE** 8192,6656 PLUS D 1a
   SAVE d1"+SYS 2A"**CODE** 8192,6656 PLUS D 2a

You should now have the correct system file on your disc.

COPYING THE COMPILER FILES

You now need to copy the compiler programs and data files. This is not as simple as it would seem, as you cannot use the **SAVE** command to copy OPENTYPE files. In addition, you will need to change the AUTOLOAD program. This comes without a LINE number for auto-running, so **LOAD** and **LIST** it. It should look like this:-

```
  10 REM LOADER FOR HISOFT C1.3
PLUS PATCH TO ADD HOOK CODE #46
TO DISCIPLE VERSION 3B
  20 LOAD d*"cc.scr"SCREEN$
  40 POKE @665,135: POKE @666,2      delete this line
  60 LOAD d*"cc.code"CODE
  70 LOAD d*"patch.code"CODE
  80 BORDER 7: INK 1: PAPER 7:"
CLS: POKE 23695,PEEK 23693
  90 POKE @6,0: POKE @8,1            printer setup
 100 RANDOMISE USR 25200
```

As you should by now have a system at higher level than 3b, you don't need line 40. If you have a PLUS D, you REALLY don't need line 40 - it zaps out instructions in the system file.

The second thing to do is to fix the printer handling. The standard library files come with carriage return and line feed at the end of each line. The inbuilt text editor adds only a line feed. Depending on exactly how you have your printer set up you may need to **POKE @8** either zero or one. Try it as distributed and see whether it works, if it doesn't, change it.

With line 40 deleted, put your new disc in drive 1 and then type:- **SAVE** d1"AUTOLOAD" **LINE** 20 - you now have the loader set up.

The CODE files can simply be copied from the distribution disc to your own disc. The commands you use will depend on whether you have a one- or a two- drive system. Assuming you have a one-drive system enter:-

```
SAVE d1"cc.scr" TO d1
SAVE d1"cc.code" TO d1
SAVE d1"patch.code" TO d1
```

You will, of course, have to swap discs over several times. Hence the advice to write-protect your distribution disc.

At the end of this you should have a working system. Your disc should have the following file on it:-

```
1  system code file (name depending on your interface)
2  AUTOLOAD       BAS    20
3  cc.scr         SCREEN$
4  cc.code        CDE 25200,25600
5  patch.code     CDE 26998,808
```

If you don't, check what you did to set up the disc.

Library files

The OPENTYPE library files can only be copied using either the MOVE command or the inbuilt text editor. I suggest you use the text editor.

Place your new system disc in drive one and re-boot your machine from it. You should get the logo screen and the startup as described in the Hisoft documentation. If you don't, check the disc carefully.

Assuming you have got the compiler loaded, you now need to copy the library files. To do this, use the inbuilt editor. The EDIT key switches you into the editor (check the Hisoft documentation on this). Once in the editor,

1. Insert the distribution disc and type:- g,,1:stdio.h

2. Insert your new disc and type:- p1,9999,1:stdio.h
   d1,9999

3. Insert the distribution disc and type:- g,,1:stdio.lib

4. Insert your new disc and type:- p1,9999,1:stdio.lib
   d1,9999

The library texts are now copied. I suggest that you try to print one of them, as follows

5. g,,1:stdio.h
   w1,9999

You may well get this double spaced. As mentioned above, the library files include both CR and LF at the end of each line. If you get double spacing, this is probably for the best, as it means that your own files will probably be single spaced, as the editor includes only an LF at the end of each line.

USING THE COMPILER

I'm not going to say too much about the compiler and how to use it in this article. As a compiler, it works fine. There are some idiosyncrasies and limitations, as with all compilers, but once you get to know them, you can live with them. The ones that trouble me most are:-

1. When you have a text file loaded, you must include it for the compiler to read it and at the end, enter symbol shift-I for and end-of-file marker. This is all carefully and clearly described in the Hisoft manual. I still find it confusing.

2. **SAVE YOUR TEXT BEFORE YOU COMPILE IT** (did you get that, if not, read it again) save your text before you compile it. If your program is correct, it will compile correctly. However, I have yet to write a correct program and there are some source errors which will cause the compiler to try to read its own navel. When it does, you lose your text file.

3. The #translate command does not work when you try to save to disc. I have asked Hisoft whether they will let me have a look at the code to see whether I can fix it (the changes to use OPENTYPE which I did for them are in patch.code) but I hold out no guarantees. It should be possible to #translate on to tape, then to copy the CODE file to disc, but my tape recorder is broken and I haven't saved anything to it for years, so I haven't checked out this route.

Copyright

Hisoft permit only one working copy. The above process will produce you that one working copy. Additional copies should not be produced.

The next step is to try it out.

Sample Program

Finally, a sample program to print a table of characters on the print output.

```
#include 1:stdio.h
int a, b, c, d, e;
char nl;
FILE *fp,
     *sc=0, *pr=3;
main()
 ( /* this is the curly bracket on the F key */
 fp=pr;
 nl=13;
 for(a=1;a<256;a++)
  ( /* F key as above, and the slashes are those on D */
  fprintf(fp,"%c%3d 0x%02x //%03o",nl,a,a,a);
  if( 32<a && a<128 )
   fprintf(fp," %c%,a);
  ) /* curly bracket on the G key */
  fprintf(fp,"%c",nl);
 ) /* G key as above */
#include ?1:stdio.lib?
```

If you can run this successfully and also list it on the printer, then your setup is working.

I hope to be back soon with a short series on the C language.

LLISTER

By: Ted Bond.

When you require a listing on a full size printer, it is more than likely that the printer parameters needed do not correspond with the printer's defaults nor with those set in the program. If you use a Serial printer with Interface One, then to list a program uses channel "t" whereas most other printing needs channel "b". Two program listings follow, one for Serial printers and one for Parallel, both designed to alleviate the problem.

PARALLEL

```
9991 STOP
9993 POKE @6,1: LPRINT CHR$ 27;"M": INPUT "How many characters per l
ine? "; LINE x$: POKE @5,VAL x$
9995 INPUT "How many lines per inch (6/8) "; LINE x$: LPRINT CHR$ 27
;("0" AND x$="8")+("2" AND x$="6")
9997 INPUT "Enter date "; LINE a$: INPUT "Enter heading or press ""E
NTER""  "; LINE b$: INPUT "How many copies? "; LINE x$
9999 POKE @6,0: FOR y=1 TO VAL x$: LPRINT a$'b$: LLIST : LPRINT CHR$
 27;CHR$ 12: NEXT y: PRINT #0;"DONE"
```

SERIAL

```
9991 STOP
9993 CLEAR #: OPEN #4,"b": FORMAT "b",9600: OPEN #5,"t": FORMAT "t",
9600: PRINT #4;CHR$ 27;"[5w": INPUT "How many characters per line? "
; LINE x$: PRINT #4;CHR$ 27;"[";VAL x$;"s"
9995 INPUT "How many lines per inch (6/8)? "; LINE x$: PRINT #4;CHR$
 27;("0" AND x$="8")+("2" AND x$="6")
9997 INPUT "Enter date "; LINE a$: INPUT "Enter heading or press ""E
NTER""  "; LINE b$: INPUT "How many copies? "; LINE x$
9999 FOR y=1 TO VAL x$: PRINT #5;a$'b$: LIST #5: PRINT #4;CHR$ 12: N
EXT y: CLEAR #: PRINT #0;"DONE"
```

In use, the appropriate routine is merged with the program to be listed and then started by "GOTO 9993". Prompts will appear and a listing be produced. The writer uses lines 1 to 9 in his normal programs for REMs which describe the program and its state of development. If those lines are not used, the LLIST programs may be renumbered using lines 1 to 9. If this is done, the "STOP" in the first line may be removed but a "STOP" must be placed at the end of the last line, 9 after renumbering. The command in the present line 9999 "LLIST" ("LIST #5") must be replaced by "LLIST 10" ("LIST #5, 10"). If you do not renumber, the LLIST program will itself be listed at the end of the desired listing, but the writer finds this a small price to pay for the convenience of the program.

The Parallel program is customised for a 48k Spectrum used with a DISCiPLE or PLUS D Interface and an NEC P2200 printer. The Serial program is used with a 48k Spectrum, Interface 1 and a Mannesmann Tally MT160 printer with a Serial interface built in.

Both programs will require amending to suit the readers own printer and interface and the relevant manuals will help. In line 9993, "CHR$ 27;"M"" ("CHR$ 27;"[5w") select 12 characters per inch. You may prefer 10 c.p.i. or to insert a prompt to offer yourself a choice. In the Parallel program, "POKE @6,1" and "POKE @6,0" respectively allow and disallow printer control codes to pass through the PLUS D, and POKE @5, VAL x$ sets the line length via the disc operating system, rather than direct to the printer.

In the Serial program, in line 9993, you may have to alter the Baud rate to suit the needs of your printer.

Finally, the "STOP" in line 9991 is there as a precaution. From time to time, the writer carelessly saves a program after, rather than before, llisting with the result that the LLIST program is stuck on the end. When the program is next loaded and used, the consequences would probably be most unexpected were the "STOP not there!

Happy LLISTing!

= + = + = + = + = + = + = + =

ADVENTURE CORNER

By: Paul Rigby.

William Crowther and Don Woods are universally credited with creating the first adventure back in the late-1960's, thus spawning a whole new field of computer leisure entertainment. Since that time there have been countless copies of that classic produced in the USA and in this country. But who were Crowther and Woods and how did their first adventure evolve?

Many adventure enthusiasts believe or assume that the first, or Original Adventure, was conjured from the fertile imagination of William Crowther. This is not strictly correct. In fact Mr.Crowther, who graduated from the University of Massachusetts, in 1953, with a BSc in Physics, had a closer relationship to caves than many people think. He actually worked for the Cave Research Foundation (CRF). In the late-sixties the CRF were conducting scientific research and producing maps of caves for the National Park Service in the USA. During his four year service, 1963 to 1967, with the CRF William Crowther spent much of his time in Kentucky mapping one particular cave - Mammoth Cave. Anyone who is looking for the inspiration for the Original Adventure need look no further because the first map for Original Adventure matches part of Mammoth Cave.

But why create the game at all? Well, it was chiefly constructed for the purposes of entertaining his two children Laura and Sandy Crowther using his talents with the DEC PDP-10 computer in the office at work. While he was working for a consulting firm in Cambridge (USA), in the research team, he dabbled with the DEC and began to write the Original Adventure in Fortran (A language which is geared towards scientific research, formulas and so on). To actually go ahead with this task he not only had the interest in caves to drive him on but an interest in the world of Fantasy. He often played a basic form of role-playing game with his children, called "Mirkwood", which was vaguely similar to the Dungeons and Dragons system, using figures, associated scenery and a book of rules. Having enjoyed this, Crowther considered linking this form of entertainment with his knowledge of computers and caves.

The actual programming of, what was to be called "Adventure", was done at weekends and took about a month to complete. This, first version, was not the complete Original Adventure that is known today but it did include the heart of the final version such as that pesky, axe throwing dwarf, the snake/bird puzzle and some of the magic words such as "xyzzy". By the way, if you have played any version of the Original Adventure and are rather mystified at the reference to "spelunking" in one part of the game the explanation is that it comes from the word "Speleology". Which, basically, means the scientific study of caves.

The Crowther children began to play the "finished" adventure at home, via a teletype and modem, and at their father's office whenever possible. However, word soon spread about this new computer program, which worked in a similar way to many of todays utility driven adventures (See last month's Adventure Corner), which resulted in the whole office racking their brains trying to solve the puzzles of the cave! Crowther placed the adventure on an early form of computer network called ARPAnet which was used by the company he worked for. Soon users from all over the country had discovered the Original Adventure.

One of those users was a gentleman named Don Woods.

Studying computer science and electrical engineering at the University of Stanford, Don Woods frequently logged on to ARPAnet while he was attached to the Stanford Artificial Intelligence Laboratory in 1976-77. A colleague in the medical centre, at Stanford, found "Adventure" on the ARPAnet system and told the 23 year old Woods. After a casual look at the game Woods became enthralled by this new concept of computer gaming. On examining the adventure he stumbled across a number of bugs which he thought could be fixed to improve the game. A number of new ideas were also contemplated by Woods which, he believed, could be incorporated within the adventure. All he had to do was find Crowther and discuss those ideas with him. Which was a problem because he did not have a clue where Crowther lived or worked!

Don Woods decided that the only way to find Crowther was to send a message, via ARPAnet, to every host site and hope that Crowther would see it. Woods made the big assumption that Crowther was still using ARPAnet considering the Original Adventure had been programmed almost 10 years ago! As luck would have it Crowther did receive the message and immediately contacted Woods. The two finally agreed to enlarge the basic code using some of Wood's ideas. Crowther sent a revised code to Woods, which mainly differed from the original by having "hooks" within the program which allowed additional pieces of code to be inserted.

Re-writing of "Adventure" was aided by ideas from Wood's friends and colleagues at Stanford, such as Bob Pariseau, who suggested the inclusion of extra treasure and a points routine to gauge progress. The final version being completed in April of 1976 or 1977. Woods then placed the improved version on the Stanford computer. Complaints were then recieved. Not from Users criticising "Adventure" but from the computer operators at Stanford who complained to Woods that the system was overloaded and strained with users trying to log-on to "Adventure"! Woods continued to improve the adventure making it bigger and more exotic such as the inclusion of the volcano and the associated puzzles with the Giant's Room. Consequently, Woods bowed to pressure from enthusiasts making the revised version of the adventure generally available.

Because the Original Adventure was, basically, Public Domain there were no problems with copyright if a computer programmer wished to construct his own version of the Original Adventure. The large computer software organisation, Microsoft (who have created MSDOS for PC's, the famous Fight Simulator Ver.1,2 & 3, and so on.) was one of the first to create their own version. This version of Original Adventure soon became, as is a habit with Microsoft, the Industry Standard. Which must have brought a smile to the lips of Crowther and Woods who probably did not consider the concept of an "Industry Standard" adventure! However, Microsoft's move to take this form of computer software seriously, helped to establish the adventure as a serious and viable undertaking. People, who had not previously looked at such "ramblings fit for hackers" quickly changed their views and their outlook towards the adventure. Microsoft Adventure began to appear on the first personal computers such as the IBM PC and the Apple II.

When the Original Adventure reached the shores of Great Britain it was snapped up by many adventure authors as the base for their "new" game. Many versions of Original Adventure have, therefore, been thrust onto the unsuspecting adventure player. Not all of them, unfortunately, were very good. In fact it reached a point were adventure reviewers, in computer magazines, were screaming for something new to review instead of another version of Original Adventure! Nowadays, such a release is very rare.

However, if you have never played a

version of Original Adventure I urge you to make a point of doing so, for its historical value as being the first adventure, if nothing else. So which one to play? Well, there were three very good versions produced. Two of which are still generally available. Classic Adventure was Melbourne Houses's, text only, attempt. This is the oldest of the three and probably the hardest to find, it is also the least polished version. Serf's Tale is the most recent version and probably the cheapest. It was published on the Player's label and sells for £1.99. It is a text only game and incorporates many twists to the original plot - highly recommended. The last of the three, and the most elaborate, is Level 9's Colossal Cave. It is elaborate for two reasons. Firstly, it incorporates graphics, which were, wrongly, maligned by some users. Secondly, it is only generally available as part of a three-pack adventure compilation called The Jewels of Darkness, along with two other Level 9 adventures, Dungeon Adventure and Adventure Quest. Level 9 have added an "end-game" to their version which extends the, already, large Original Adventure making it even bigger! It is the most expensive version to buy, because it comes with two other adventures (both classics in their own right) priced, originally, at £15.00 but you should be able to buy it via mail order at about £7.00 to £8.00.

If you intend to purchase any of the above then I wish you luck and, most of all, enjoy yourself! Next month I may have the latest version of the PAW by Gilsoft to review. Apart from the program upgrade, Gilsoft are offering a suite of three additional programs which should help adventure authors using the PAW. A review? In Adventure Corner? Whatever next?

FILOFAX PAGES

By: Carol Brooksbank.

Once you have fallen victim to the dreaded filofax your life becomes unmanageable without it, and it is always annoying that they never print the pages that would be exactly right for your particular job or hobby.

This program lets you design and print your own pages, in exact filofax size and format, with the positions of the holes marked ready for punching. You need Beta Basic and my own 'SMALL IS BEAUTIFUL' printing routine which appeared in FORMAT in December 1988 (Vol 2 No 5).

Program 1 is the listing for a plain filofax page. You can see from the line numbering that it is in three parts, each separated by a SAVE....SCREEN$ instruction. When you run the program it will draw each of the three screens that make up the full filofax page and save them as "Fax1", "Fax2" and "Fax3". The SAVE could be to any storage medium, even tape, as only the top 22 lines of the screen are needed (so the usual tape saving messages wont corrupt the final print).

The listing is written using Beta Basic version 4.0. If you are using an earlier version, omit line 10 and all the lines starting FILL USING. The only difference will be that the punch holes will be marked by plain circles instead of black dots. If you don't have Beta Basic at all, you can rewrite it to work (very slowly), in ordinary BASIC, using the method Andy Wright gave in his article "The Rose", see FORMAT Dec'88, to overcome the lack of a DRAW TO command.

The three screens are printed using 'Small is beautiful' to produce the page. (Fig 1). You should MERGE the following lines into the BASIC I gave you with that program.

```
 40 LET mode=2: LET dots=2
 50 LET margin=0
 80 REM
102 FOR z=1 TO 3
105 LET s$="Fax"+STR$ z
107 LOAD d1;s$ SCREEN$
120 FOR Q=1 TO 21
165 NEXT z
```

NOTE: mode 2 is the same as ESC Y for those who do not have the ESC "*" control.

You can SAVE this amended BASIC as "FILOPRINT" for future use, if you wish. If you want to get two pages side-by-side on one sheet of paper, you can put the paper through again for the second page, with

line 50 changed to:-

```
50 LET margin=41
```

This blank page is the framework upon which you design your personalized pages. The lines and text for each 'third' of the page are entered into the unused lines leading up to each SAVE instruction. Beta Basic's CSIZE will give you a variety of small, neat typefaces.

My particular interest is machine knitting, and Fig 2 shows you my filofax pages for keeping notes about people's measurements and the yarns and tensions I use. The BASIC for this is produced by entering the three blocks of basic lines given in Program 2. These lines should be typed in and then MERGEd with those in Program 1. This should give you an idea of how to use the program to produce your own pages.

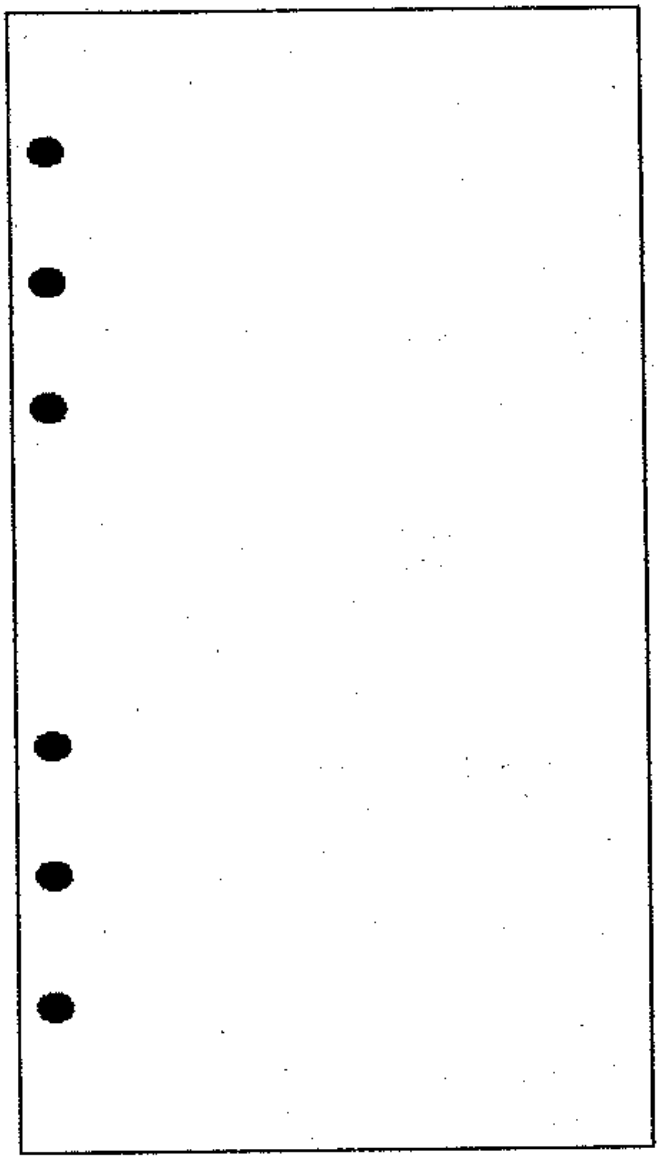


Fig 1.

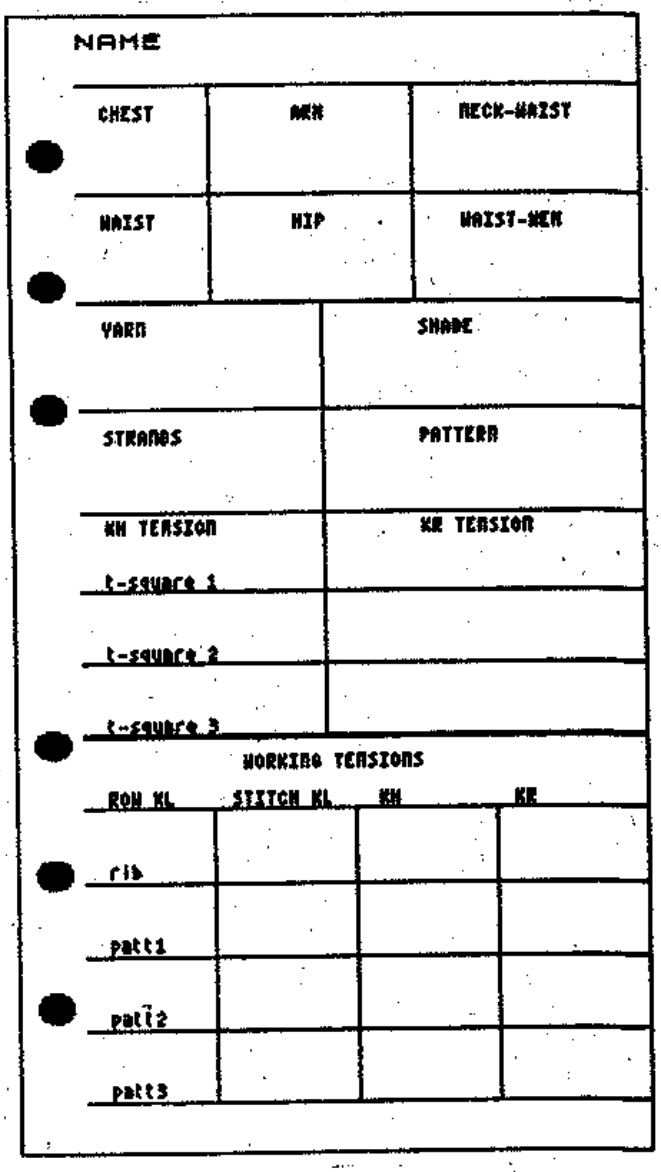


Fig 2.

Who knows - if everyone at the bird-watching club or the train-spotting society needs to make the same sort of notes, you may be able to make a lucrative sideline out of printing filofax pages.

HAPPY FILOFAXING....

PROGRAM 1.

```
  10 LET a$=STRING$(32,CHR$ 255)
  20 PLOT 0,8
  30 DRAW TO 0,175
  40 DRAW TO 224,175
  50 DRAW TO 224,8
  60 CIRCLE 13,113,6
  70 FILL USING a$;13,113
  80 CIRCLE 13,55,6
  90 FILL USING a$;13,55
1999 SAVE d1"Fax1" SCREEN$
2000 CLS
     PLOT 0,8
2010 DRAW TO 0,175
2020 PLOT 224,8
2030 DRAW TO 224,175
2040 CIRCLE 13,168,6
2050 FILL USING a$;13,168
2060 CIRCLE 13,19,6
2070 FILL USING a$;13,19
3999 SAVE d1"Fax2" SCREEN$
4000 CLS
     PLOT 0,8
4010 DRAW TO 0,175
4020 PLOT 0,8
4030 DRAW TO 224,8
4040 DRAW TO 224,175
4050 CIRCLE 13,130,6
4060 FILL USING a$;13,130
4070 CIRCLE 13,72,6
4080 FILL USING a$;13,72
5999 SAVE d1"Fax3" SCREEN$
```

PROGRAM 2.

BLOCK-1

```
 100 CSIZE 8,8
     PRINT AT 1,3;"NAME"
 110 PLOT 24,144
     DRAW TO 224,144
 120 CSIZE 4,8
     PRINT AT 5,8;"CHEST";AT 5,25;"ARM
";AT 5,40;"NECK-WAIST"
 130 PLOT 24,96
     DRAW TO 224,96
 140 PLOT 71,144
     DRAW TO 71,48
     PLOT 143,144
     DRAW TO 143,48
 150 PRINT AT 11,8;"WAIST";AT 11,25;"H
IP";AT 11,40;"WAIST-HEM"
 160 PLOT 24,48
     DRAW TO 224,48
     PLOT 24,47
     DRAW TO 224,47
 170 PRINT AT 17,8;"YARN";AT 17,36;"SH
ADE"
 180 PLOT 110,48
     DRAW TO 110,8
```

BLOCK-2

```
2080 PLOT 110,175
     DRAW TO 110,23
2090 PLOT 24,168
     DRAW TO 224,168
2100 PRINT AT 2,8;"STRANDS";AT 2,36;"P
ATTERN"
2110 PLOT 24,122
     DRAW TO 224,122
2120 PRINT AT 7,8;"KH TENSION";AT 7,36
;"KR TENSION"
2130 PRINT AT 10,8;"t-square 1"
2140 PRINT AT 14,8;"t-square 2"
2150 PRINT AT 18,8;"t-square 3"
2160 PLOT 24,87
     DRAW TO 224,87
2170 PLOT 24,55
     DRAW TO 224,55
2180 PLOT 24,23
     DRAW TO 224,23
     PLOT 24,22
     DRAW TO 224,22
2190 PRINT AT 20,20;"WORKING TENSIONS"
```

BLOCK-3

```
4090 PRINT AT 1,8;"ROW KL";AT 1,19;"ST
ITCH KL";AT 1,31,"KH";AT
     1,44;"KR"
4100 PLOT 24,159
     DRAW TO 224,159
4110 PRINT AT 5,8;"rib";AT 9,8;"patt1"
;AT 13,8;"patt2";AT 17,8;
     "patt3"
4120 PLOT 24,126
     DRAW TO 224,126
4130 PLOT 24,94
     DRAW TO 224,94
4140 PLOT 24,62
     DRAW TO 224,62
4150 PLOT 24,30
     DRAW TO 224,30
4160 PLOT 70,159
     DRAW TO 70,30
4170 PLOT 120,159
     DRAW TO 120,30
4180 PLOT 170,159
     DRAW TO 170,30
```

SUPER BORDER

By: Terry Simpson.

This is a short routine that I have been using for some time to put an eye-catching border on screens. I often run stalls at fund raising events for the local scout group and it enables me to grab peoples attention as soon as the come into the hall.

It works by calling a small machine code routine thats stored in the 48k printer buffer. It sets up a border around a selected number of lines by setting character squares to flashing using the selected INK and PAPER colours. You should build this into your own programs. Try it out, I think you will like it.

```
   1 REM DISPLAY BORDER ROUTINE.
   5 DEF FN B(I,P,L)=USR 23296
  10 IF PEEK 23296<>221 THEN  GO SUB 1000
  20 INPUT "Ink Colour=";I'"Paper Colour=";P'"Number of lines=";L: I
F L<2 OR L>24 THEN  GO TO 20
  30 LET X=FN B(I,P,L)
  40 STOP
1000 REM set up machine code.
1010 FOR I=23296 TO 23390: READ N: POKE I,N: NEXT I: RETURN
1050 DATA 221,42,11,92,221,94,4,221,70,12,62,0,128,40,7,123
1060 DATA 198,8,16,252,24,1,123,198,128,79,221,70,4,221,94,12
1070 DATA 62,0,128,40,7,123,198,8,16,252,24,1,123,198,128,33
1080 DATA 0,88,6,16,119,35,113,35,16,250,245,221,70,20,62,20
1090 DATA 184,48,2,6,20,241,203,56,30,31,22,0,113,25,119,35
1100 DATA 119,25,113,35,16,246,6,16,113,35,119,35,16,250,201
```

TASCALC REVIEWED

By: John Wase.

TASCALC: A 128K SPREADSHEET

Unlike the venerable Omnicalc, Tascalc is intended only for 128K models, either on +3 disc (£19.95) or cassette, for the 128 and +2, with facilities for microdrive transfer (£17.95). The spreadsheet (which will hold 64K of data) is entered from an opening screen: beware; Basic is overwritten and once entered, there is no return, you have to reset your computer to get back to Basic. The main screen displays rows (numbered in inverse on the left) and columns (identified by letter), an abbreviated help screen (called "prompt") above, and a further prompt line below: the screen is a window which scrolls over the 157 row x 32 column display, all in Tasman's inimitable 64-column characters.

Again unlike Omnicalc, the 128K memory allows you to use all the rows and columns at once. Cells or groups of cells are readily identified, formulae are easily entered, and text, too, with a little fiddling: one line per cell, though displaying and printing can present problems. The number of significant figures displayed can be changed at will: numerals can also have various prefixes and numerals and text have various aligning features. Calculation is, by Spectrum standards, tolerable, and you can clear (Zap) the spreadsheet at the end.

Unfortunately, getting the information out is not so easy. You can print the spreadsheet: the driver has facilities for inputting printer control codes for the heading and for the main sheet. You can do graphs, but can't print them, only save them. Although switching between bar-chart and line graph is quite fast, the graph routine itself is pretty minimal; it merely draws integer ticks on the X axis (the labelling routine allows only one row of text below this) and it allows no negative y values at all. You can save data as arrays, but you can't export ASCII code and it is difficult to set up the spread sheet to do something and put in sequential batches: data merging is impossible and loading clears the lot. So you can't incorporate it in Tasword. ASCII file export and a separate graph drawing program would have improved the facilities enormously.

For proprietary disc users, the worst feature is that the cassette version is all in code, if only they had put the graph features in a separate program there would have been room for some basic control lines. Although the microdrive version saves to a +D disc, trying to save data to disc or a SCREEN$ (those graphs) crashes the machine. If enough users are interested, Bob Brenchley tells me that the user group will look at it: until then, back to cassette.....

Tascalc, from Tasman Software, Springfield House, Hyde Terrace, Leeds, LS2 9LN. Phone 0532 438301.

INSIDE G+DOS

By: Stephen Warr.

This sees the start of what will hopefully become a regular series all about G+DOS, how it works, what it contains and how you can use parts of it in your own programs.

The series will probably be the most helpful to PLUS D owners who know at least a bit about machine code, but in later articles I will be including several example routines that less experienced readers should find useful. The overall aim of the series is to teach you the information you need to expand the G+DOS system by yourselves.

Starting from very basics, the PLUS D's G+DOS (Disc Operating System) contains 8K of ROM followed by 8K of RAM. The ROM runs from address 0 to address 8191 and contains all the routines that control the disc and printer ports directly - the routines that actually read or write bytes of data. The RAM runs from 8192 to 16383, ie. to just below the start of normal RAM and the beginning of the screen data at 16384. It contains the necessary syntax checking for the new disc commands and the ROM calls needed to execute them. The only exception is the POKE @ command which is held completely in ROM.

The problem for us is that most of the time G+DOS just isn't there, it is hidden behind the normal Spectrum ROM and most of the time acts as if it didn't exist. Because of this, it is known as a 'shadow' ROM (except that half of it is actually shadow RAM, but don't worry about that too much). The PLUS D only appears, or 'pages' itself in, if the normal ROM jumps to one of 4 addresses. When this occurs, the normal ROM temporarily disappears and is replaced by the code held in G+DOS.

The PLUS D has no networking, unlike the DISCiPLE, so there is still quite a lot of space left in both the ROM and the RAM. Obviously we cannot use the ROM, but with over 2K of spare RAM, where better to put extra disc routines? This has the advantage of leaving the normal 48K/128K of RAM untouched and so this is where I will be placing routines in future articles.

The first 167 bytes of PLUS D RAM contain various system variables, tables of data, and a number of addresses which are called from the ROM at various points. The POKE @ command affects these bytes directly so that POKE @0,7 means poke the first byte of G+DOS RAM (at 8192) with 7. The table on the next page gives a complete list of this area - some you may find useful, others not. For each entry, the first column gives the displacement from the start of G+DOS RAM, ie. the first number in a POKE @ command. The second column gives the number of bytes used in that particular block of data, and the third gives a short desciption of its use and the values allowed. The last few entries are particularly useful as they provide a way of directing the ROM to our own routines in RAM.

PLUS D (G+DOS) INTERNAL SYSTEM VARIABLES

DISP	BYTE	DESCRIPTION
0	1	Determines how much the border flashes during disc operations. 0=No flashing, 7=Maximum flashing.
1	1	Format capacity of drive 1. Value is the number of tracks on disc (add 128 if the disc is double sided).
2	1	Format capacity of drive 2. As above, or zero if no drive 2 present.
3	1	Drive stepping rate in milliseconds. Values 6-255.
4	1	Unused by the PLUS D.
5	1	Number of characters on each printer line.
6	1	G+DOS outputs absolute codes to the printer when this is zero, otherwise keywords will be expanded, etc.
7	1	Line feed spacing in 1/72's of an inch.
8	1	Number of line feeds output after each carriage return Usually 0 or 1.
9	1	Character width of left hand margin.
10	1	Poke this with zero if "#","£" and "(c)" are not to be printed in bit image graphics mode.
• 11	1	Value is zero if you want to use the parallel printer port. Otherwise, the ZX printer can be used.
12	2	Unused by the PLUS D.
14	2	Jump to the address held here for extended syntax checking, ie if the command currently being edited or run is neither a normal Spectrum command nor a G+DOS command then you can direct the ROM to wherever you want, allowing you to make up your own commands. The shadow ROM will NOT be paged in, and the 'A' register will contain the code of the first character in the BASIC statement.
16	2	Usually holds the value 8335 (#208F). SEE DISPLACEMENT 152.
• 18	8	This is a string of printer control codes (up to a maximum of 8),used to reset the printer. The byte 128 should follow last code if less than 8 used.
26	8	Printer codes to select the default print pitch.
34	8	Printer codes to give n/72's of an inch line spacing.
42	8	Printer codes to print single density 8 bit graphics.
• 50	8	Optional printer codes output when the system file is booted up. Usually empty.
58	8	Data for bit image graphics "£".
66	8	Data for bit image graphics "#".
74	8	Data for bit image graphics "(c)".
82	8	Printer codes to print 576 bytes of data in bit image graphics mode at 72 dots per inch. (Used by SAVE SCREEN$ 2).
90	9	SAVE SCREEN$ 2 colour pattern data. Each bit on screen is converted to a patterned block of 9 (3x3) dots on paper. Bits 7 of all 9 bytes of data (1 bit from each byte) correspond to the pattern for white, bits 6 to yellow and so on. The bits from the first 3 bytes of data give the top row of the pattern, the next three give the middle row, and the last 3 bits give the last row of dots.
99	2	This holds the address to jump to the handling routine for outputting the next byte to the centronics port.

		It is usually set to 5161 (#1429), but is changed if the present byte is a data byte following a TAB/AT control.
101	1	Used as a printer variable, but only bit 1 is used. When the bit is set the next carriage return printed will instead be ignored.
102	2	G+DOS error return address. Used by the snapshot and Command Code routines, the ROM will jump to this address rather than print an error message. NB. the format message "Are you sure", and the "Overwrite?" message are both treated as error messages.
104	20	This is the snapshot catalogue entry data, or 'header' data, used when making a snapshot. The first byte is loaded with 5 for a 48K snapshot, 7 for a SCREEN$ and 9 for a 128K snapshot. The next 10 bytes give the name (usually "Snap", but can be changed).
124	4	Unused by the PLUS D.
128	3	G+DOS calls here before executing SAVE SCREEN$ 1. Usually holds a RET instruction.
131	3	G+DOS calls here before executing SAVE SCREEN$ 2. Usually holds a RET instruction.
134	3	Calls here before outputting a byte to the printer. Usually holds a RET instruction.
137	3	Calls here before executing a POKE @ command. Usually holds a RET instruction.
140	3	Calls here after the system file has been loaded, but before the centronics port is initialised. NB. it is also called by the interrupt routine, ie. 50 times a second. Usually holds a RET instruction.
143	3	Usually holds a RET instruction. SEE DISPLACEMENT 152.
146	3	Calls here to load the "Auto" file. Usually holds a JP 10478 instruction.
149	3	Calls here to transfer the header data of "Auto" file before searching the catalogue to see if the file exists.Exits if it doesn't or calls the above address to load the file. Usually holds JP 12171.
152	3	This address is called 50 times a second by the interrupt routine. It usually holds a JP 8773 instruction where there is a short routine that picks up the value at 8208 (displacement 16) and jumps to the address it holds. This usually directs it to 8335 (displacement 143), where a RET instruction returns execution to the ROM.
155	3	Holds a jump to a routine that prints the G+DOS version number after the system file has loaded.
158	3	G+DOS calls here to jump to the Command Code handling routine. Usually holds a JP 8846 instruction.
161	3	This is called continuously while waiting for a key to be pressed after the snapshot button has been pushed. Usually holds a JP 8469 instruction.
163	3	Calls here before the BASIC commands are run/syntax checked (excluding the POKE @ command). Usually holds a JP 8359 command.

Thats all for this month. Next time I will be explaining how the PLUS D pages itself in, and investigating the mysteries of the disc directory.

YOUR LETTERS

STAR*LETTER* *STAR*LETTER

Dear Editor,

I have just taken delivery of the MGT TWOFACE. One of the reasons for buying the interface was to enable me to continue to use the Alphacom 32 printer at the same time as the PLUS D. I was therefore dismayed to hear that MGT had experienced problems in using the Alphacom.

As I soon discovered for myself, on Switching from the PLUS D to the Alphacom the printer would not work unless the Spectrum was reset. For most applications this was unacceptable.

However, on investigation I found the cause of the problem. When the PLUS D pages in it overwrites the printer channel information with its own pointers. When switching to the Alphacom the addresses do not revert to normal. Therefore to overcome the problem all that is needed is to make the following pokes:-

```
POKE 23749,244
POKE 23750,9
POKE 23751,196
POKE 23752,21
```

I hope this will help other readers who may experience similar problems.

Yours Sincerely, R.W.Bray.

Dear Editor,

Could I please use your pages to pass on to your readers the news that RAMMPACK has a new organizer. RAMMPACK was formed two years ago as an independent user group for the Ram Music Machine and has been very successful in attracting some 150 members.

From now on it will be known as the SPECTRUM MUSIC GROUP and we aim to expand to cover other interfaces including the XRI, the Cheetah Specdrum, and Midi interfaces.

If anyone who is interested would like to send a stamped addressed envelope to us we will be pleased to send them more details.

Sean Sanderson,
SPECTRUM MUSIC GROUP,
'Chesters',
Chesters Lane,
High-Bentham,
Via Lancaster,
LA2 7AN.

Dear Editor,

I recently purchased the latest version of the Disc Manager from Better Bytes of Gosforth. I would like to congratulate Dave Hood on producing such a great program. It must be one of the most useful programs on the market. I would say to all disc users 'Your disc collection is incomplete without this program and its easy to follow manual'.

Well done Dave, I look forward to your future releases.

Yours Sincerely, Bill Scully.

Thanks for your letter Bill, its one of many that I have received, over the time Better Bytes have been advertising with us, all full of praise for Disc Manager. I can't print them all - there just isn't room. But I like to receive them anyway. And its not just Better Bytes either. We seem to have attracted adverts from some of the best companies still producing for the Spectrum. So remember, support our advertisers, because they support you. And if you see a company advertising elsewhere, ask them why they dont advertise in FORMAT. ***Ed.***

Letters printed may be edited for length or clarity. The writer of each months STAR LETTER wins an EXTRA 6 months subscription to FORMAT.

FASTFILE

By: Clyde Bish.

"All this memory saving for adventurers is all very well, but what about the rest of us!" I hear INDUG enthusiasts shout. Well, here's a memory saving routine just for you (although I suspect the adventurers may find a use for it as well!) It's called "FASTFILE" and is a system for holding information, in any form, in a DIMentioned string of some 40,000 plus characters, with a machine code routine, searching at some 50,000 characters per second to extract from it the information required and print it to screen or produce hard copy.

I wrote it originally as an index to Dartmoor Letterboxes - a strange past time we have in Devon of searching for rubber stamps hidden under rocks on Dartmoor! Since then the program has been used as an index system to stamp collections, telephone numbers, and even seasonal hymns by the local vicar! So without further ado let's get to work.

Use the machine code loader (Program 1) to enter the numbers from Table A, reading across each line. Check each entry as it appears on screen, make a note of any mistakes, and correct them at the end with:- POKE address,correct number.

Now NEW the machine. (Panic not! Your code is safe above RAMTOP) Next, type in Program 2. This sets up variables used by the system. Once in, RUN the program, then delete the one program line by simply entering 10 and pressing RETURN. This may seem a bit crazy, but, readers of my earlier articles will be nodding sagely, and saying "Ah yes. It's safely stored in the Variables Area". If you don't believe me type **PRINT N** and press ENTER! See, I told you! For the previously uninitiated, we need all the space we can get for the file so once a variable is set there's no point in keeping the program line that set it.

Now type in Program 3. This is the main program that will be doing all the work. Note that the "STOP" in lines 1010, 2000, 2120 and 2125 are all tokens and must be entered in symbol shift mode as before. Remember the old proverb "Fools rush in where angels fear to tread" so polish up your halo and save the program and code, just in case of mistakes with **SAVE "fastfile" LINE 9000: SAVE "fastC**ODE" CODE 65368,111 and verify both parts.

Now you're ready to try it out. Type **GOTO 100** (ENTER) and the menu will appear. Each option is chosen by pressing the appropriate number. But first an awful warning. If you get an error message (and you may do as there is a minimum of error trapping included to free as much space as possible for the file) always restart with GOTO 100. Never use RUN or you'll lose all your stored information! This is what you can do:

1. **ENTRY**

This adds an entry to the file, provided there is enough space. (You're told how much space is free each time). The maximum INPUT length is about half a screen. (Remember to restart with **GOTO 100** if you get an Out of memory message). If you want to save space but avoid filling out the ends of lines with spaces to prevent word splitting, use the PRINT comma trick. For those who missed the earlier articles this is what you do. After typing the last character you want on the line get into E-Mode, hold on to the Caps Shift key, and press 6, followed by 0. The cursor will jump to the next line. (Three of these in succession would leave a blank line

within an entry at a cost of only 3 bytes!).

Make a file to experiment on using the fore and surnames of your family. After the last entry pressing just ENTER will return you to the menu.

2. **SEARCH (To Printer)**

Selecting this option will produce a print out (assuming of course you have a printer attached). It operates the same way as option 3 detailed below.

3. **SEARCH (To Screen)**

Select option 3, and answer the "Key" prompt by entering the word or phrase for which you wish to search. The machine code, which incidentally originated from the good old days of the ZX81, zips through your file, PRINTing out all entries which include that key. On completion the word END will be displayed. Pressing just ENTER will return you to the menu.

Try the following with your "name" file:-

a) Enter a forename - only that name will be displayed.
b) Try the surname - all entries with that surname will appear.
c) Try "Bloggs" (assuming that's not your name!) - just the END message will appear.
d) Try a single letter that you know is in the file ;- any entries which include that letter will appear AS MANY TIMES AS THEY CONTAIN THE LETTER. For example John Jones would appear twice in the key were J, c or h, but only once if s were entered. The moral of this exercise is that the more specific is the key used, the more selective the routine becomes. So if you were using the program as an index to magazine articles it would be better to reference spectrum programs as spl, rather than just sp, as in the latter case any entries where an s is followed by a p would be displayed.

4. **AMEND**

Now we can make some alterations to your Names file. Select option 4 and answer the "Key?; prompt with your surname. The first name in your family will appear with the prompt "Amend/Erase?". Press ENTER and continue doing so until your own name appears. Now press A and answer the prompt with a new entry - for example give yourself that bogus forename you've always wished you had. When you press ENTER you'll return to the search. When the next name appears answer the prompt with E. Nothing will appear to happen and the search will continue. Now press M to return to the "Key?" prompt. Answer it with your family surname as before and move through by pressing ENTER. When you have searched the whole file the word END appears. You will have noticed by now that your fictitous forename has been added to the file, and the name you erased is no longer stored. Press Enter to return to the prompt, then Enter again to return to the menu.

5. **SAVE**

The whole BASIC program and the variables is SAVEd. Why not just the data array? Because you also need the values held in other variables, for example the file pointer n. After VERIFYing you will be returned to the menu.

6. **LOAD**

Use this option to LOAD in an existing file for searching or updating. Existing files should only be LOADed in this way for interrogation. Don't be tempted to just LOAD in a SAVEd file directly as it will probably crash!

I would like to go into detail as to how the machine code operates but space precludes that opportunity. Suffice to say that the routine compares what you put into a$() in response to the "Key?" prompt to what is in the file array b$(), starting at the beginning and moving along until a match is found. The position of the beginning of this match is returned in variable p. The Basic program then backs from here until the marker token STOP is found, when it reverses and moves along the string, printing each character as it goes until the next STOP token is reached. Before returning the position value in p the

routine saves that position in addresses 65368 & 65369. It is from this saved position that the search continues until the end of the file is reached. As the routine always starts its search in one particular place in memory it is important that this starting place is not altered. For this reason it is imperative that the order of variables set in Program 2 is not altered and the line is RUN just as written.

To reset the file and so in effect erase all entries proceed as follows. First press BREAK when the menu is on screen. Now enter **LET n=2,** then restart with **GOTO 100.** You will now have a new blank file to use.

The information in b$() can of course be in any form and so the system could be used in any situation where fast retrieval was required. For example, in interpreting a user command in an adventure. (You see, I didn't leave you out!). A possible approach might be to store each vocabulary word in the file (which could be less than 40,000 characters) preceded by a three figure number e.g. 201, 001, 010. After each user response is made that input is broken up into separate words, each followed by a **STOP** token, then each is compared with the vocabulary in array b$(). When a match is found the three figure number associated with it is accumulated to a variable (which has initially been set the location number). This variable controls a GOSUB jump which holds a suitable response to the user command. An example is given below. You start in ROOM 1 (see Fig 1) and can move only through door ways using the command **GO** N or whatever. Type in program 4 (And make sure you also have FASTCODE on board!) and try it out using **RUN 9999.** This is how it works:-

Line 9999: This sets the arrays and variables in the correct places in the variables area. (Note that the STOPs are tokens and are entered using SYMBOL SHIFT A). Variables - f and d$ are set to response strings to save space later in the response subroutines.

Line 50: The location is set to 1 Line 55Z This is the control loop. It asks for a response (remember you can only GO N S E or W with this example!)., a space is added, then the line GOSUBs to the search routine, before calling the correct response subroutine. The variable L1 is set to L in case no move is made from the room.

Lines 9000 to 9310: This is the search routine. Each word in the user response is fed into a$() followed by a STOP token. This is compared with the vocabulary in b$() using FASTCODE. If a match is found the number associated with that word is added to L before the search is continued with the next word. When the whole response has been scanned the routine RETURNs with the correct GOSUB line number in L.

Lines 111 to 144: These are the subroutines called by the value returned in L. For example if you were in ROOM 1 and entered GO E L would return with the value 121 - 1 for the ROOM + 100 for GO + 020 for E. This would result in the response "You go E. You are in ROOM 2", and the value of L would be change to 2. What if your input was impossible (or just plain stupid!)? No match would be found and that mysterious line 19 would be called. (Of course if you have more than 19 locations you'll need to put that line down further and possibly move lines 50 and 55 also).

I'm not pretending this is a perfect system. In fact its rather clumsy but it should provide food for thought. After all my job is to provide you with ideas. Not write programs for you!

Alternatively it could be used to search for words understood in an "Elissa" type program to experiment with AI (NO! It stands for Artificial Intelligence. Computer have bugs, not cows!). I'll show you how it could be done and then you can improve on it. Have a look at program 5 as you read what follows:-

Line 800: resets variable and the arrays you've seen before.
830: R1 we need later. The program

will stop if you've said 'bye.
840: the user response goes into i$. A suitably cutting response is printed if this is a null string.
850: the input is printed to screen with a leading capital.
860-861: spaces in the input string are replaced by STOP tokens, plus one at the end. The counter n is set to string length + 1.
862: now to business! The data is RESTOREd. Look at Line 900 for a moment. You'll notice a sequence of word, phrase, phrase triplets. In essence the routine takes each response word in turn and searches for a match in k$. If one is found (i.e. p 0 AND p n) then one of the phrases READ along with k$ and held in x$ and y$ is printed. Which, is determined by R1.
882: if the search falls through the loop no match has been found so one of a sequence of general answers is given.
885: if too many general answers are given then a request is made to change the subject.
900: this is the first example of many DATA lines. You can produce the rest yourself but remember three things:-
a) the items must be in threes - keyword, phrase 1, phrase 2.
b) the order of keywords in the data list is most important. Common words must be at the end with less likely ones at the beginning or the latter will never be found.
c) my original program had a vocabulary of 100 keywords. If you want more or less you must change the number ending the FOR statement in line 862:2

To end here's an idea for an April Fool program to catch out an ardent Arcader. You need a simple (and pathetic) arcade type game on board as well as Program 5 (and of course FASTCODE!). You must begin with the line **1 POKE 236373,0** which will reset the FRAMES system variable. This is part of the machine's internal "clock" and will keep count of the time for you. You also need to include in the loop of your games program the statement:- **IF PEEK 23673 12 THEN GOTO** 800 This will monitor the time the game has been running and switch control to line 800 after about 1 minute. (You can make it longer or shorter by changing the 12, but with the game I was using boredom set in if I left it much longer!).

Now alter line 800 and add the other lines as show in Program 6. This will produce a fake "reset crash" followed by a reincarnation and on screen communication from the "Spirit of the Machine" offering psychiatric help - and having the occasional nervous breakdown! See who you can catch!

Good fooling! and see you next month.

PROGRAM 1.

```
 10 FOR f=65368 TO 65478: INPUT i: PO
KE f,i: PRINT f,i: NEXT f
```

PROGRAM 2.

```
 10 CLEAR : LET n=2: DIM a$(31): DIM
b$(39502): LET b$(1)=" STOP ": LET l=1
: LET o=0: LET m=100: LET k=23556: LET
 n$="A": LET s1=23670: LET s2=s1+1
```

PROGRAM 3.

```
 100 CLS : LET r$="": LET z=1+1: PRINT
  "File ";n$''"OPTIONS"''"1 ENTRY"''"2 S
EARCH (to Printer)"''"3 SEARCH (to Scre
en)"''"4 REVISE"''"5 SAVE"''"6 LOAD"
 110 PAUSE o: CLS : LET c$=INKEY$: IF
c$<"1" OR c$>"6" THEN GOTO m
 120 GOTO VAL C$*VAL "1000"
1001 CLS : PRINT VAL "39502"-n;" SPACE
S LEFT": INPUT "Entry?"''"(ENTER = menu
)"'' LINE e$: IF e$="" THEN GOTO M
1010 LET e$=e$+" STOP ": IF LEN e$>VAL
 "39502"-n THEN PRINT "OVERLOAD": PAUS
E m: GOTO m
1015 LET b$(n TO n+LEN e$)=e$: LET n=n
+LEN e$: IF r$="A" THEN RETURN
1020 GOTO VAL "1001"
2000 LET z=INT PI
3000 CLS : INPUT "Key? (+ENTER)"' LINE
 e$: IF e$="" THEN GOTO m
3010 LET A$=e$+" STOP "
3020 POKE VAL "65368",o: POKE VAL "653
69",o
3030 LET p=USR 65370: IF p>0 AND p<n T
HEN GOTO 3050
3040 PRINT ''"END": PAUSE o: GOTO VAL
"3000"
3050 LET p=p-1: IF b$(p)<>" STOP " THE
```

```
N GOTO 3050
3060 LET p=p+1: IF b$(p)<>" STOP " THE
N PRINT #z;b$(p);: GOTO VAL "3060"
3070 PRINT #z;'': GOTO VAL "3030"
4000 CLS : INPUT "Key? (+ENTER)"' LINE
 e$: IF e$="" THEN GOTO m
4010 LET A$=e$+" STOP "
4015 POKE VAL "65368",o: POKE VAL "653
69",o
4020 LET p=USR 65370: IF p>0 AND p<n T
HEN GOTO 4050
4030 PRINT '''"END": PAUSE o: GOTO VAL
"4000"
4050 LET p=p-1: IF b$(p)<>" STOP " THE
N GOTO 4050
4055 LET s=p+1
4060 LET p=p+1: IF b$(p)<>" STOP " THE
N PRINT b$(p);: GOTO 4060
4080 PRINT #o;"Erase, Amend, Menu"'"EN
TER to search": PAUSE o: INPUT ;: LET
r$=CHR$ PEEK k: IF r$="M" THEN GOTO m
4090 IF r$="A" THEN GOSUB VAL "4500":
GOSUB VAL "1000": CLS
4100 IF r$<>"E" THEN PRINT '': GOTO VA
L "4020"
4500 LET t=s-1: POKE 65369,INT (t/256)
: POKE 65368,t-(INT (t/256)*256)
4510 LET v=PEEK 23627+256*PEEK 23628:
RANDOMIZE (p+1+v+48): POKE 65468,PEEK
s1: POKE 65469,PEEK s2: RANDOMIZE (n-p
): POKE 65471,PEEK s1: POKE 65472,PEEK
 s2: RANDOMIZE (s+v+48): POKE 65474,PE
EK s1: POKE 65475,PEEK s2: RANDOMIZE U
SR 65467: LET n=n-(p+1-s):
PRINT ''
4520 IF r$="A" THEN RETURN
4530 GOTO VAL "4020"
5000 PRINT "Press S to save File ";n$'
"(ENTER for Menu)": PAUSE o: IF PEEK k
=VAL "13" THEN GOTO m
5005 SAVE n$ LINE m: SAVE "FASTCODE "C
ODE 65368,111: VERIFY n$: VERIFY ""COD
E : GOTO m
6000 PRINT '''"filename? (ENTER for Men
u)": PAUSE o: IF PEEK k=VAL "13" THEN
GOTO m
6010 LOAD CHR$ PEEK k
9990 LOAD ""CODE : GOTO m
```

PROGRAM 4.

```
  19 PRINT "You can't do that.": LET 1
=11: RETURN
  50 LET 1=1: PRINT r$;1
  55 PRINT ': INPUT "What will you do?
"'e$: LET E$=E$+" ": LET 11=1: GOSUB 9
000: GOSUB 1: GOTO 55
 111 GOTO 19
 112 GOTO 19
 113 LET 1=1: PRINT d$+"N"'r$;1:RETURN
 114 LET 1=2: PRINT d$+"N"'r$;1:RETURN
 121 LET 1=2: PRINT d$+"E"'r$;1:RETURN
 122 GOTO 19
 123 LET 1=4: PRINT d$+"E"'r$;1:RETURN
 124 GOTO 19
 131 LET 1=3: PRINT d$+"S"'r$;1:RETURN
 132 LET 1=4: PRINT d$+"S"'r$;1:RETURN
 133 GOTO 19
 134 GOTO 19
 141 GOTO 19
 142 LET 1=1: PRINT d$+"E"'r$;1:RETURN
 143 GOTO 19
 144 LET 1=3: PRINT d$+"E"'r$;1:RETURN
9000 LET c=1
9010 LET a$="": FOR f=1 TO 100: LET a$
(f)=e$(c): IF e$(c)=" " THEN LET a$(f)
=" STOP ": GOTO 9100
9020 LET c=c+1: NEXT f
9100 POKE 65368,0: POKE 65369,0: LET p
=USR 65370: IF p>0 AND p<n THEN GOTO 9
200
9110 GOTO 9300
9200 LET p=p-1: IF b$(p)<>" STOP " THE
N GOTO 9200
9210 LET 1=1+VAL b$(p+1 TO p+3)
9300 IF c+1>=LEN e$ THEN RETURN
9310 LET C=C+1: GOTO 9010
9999 LET n=28: DIM a$(31): DIM b$(30):
 LET b$(1 TO )=" STOP 100GO STOP 010N
STOP 020E STOP 030S STOP 040W STOP ":
LET d$="You go ": LET r$="You are in r
oom ": GOTO 50
```

PROGRAM 5.

```
 800 CLEAR : LET r=0: DIM a$(31): DIM
b$(64): LET k$=""
 830 LET r1=RND: IF k$="goodbye" OR k$
="bye" OR k$="cheerio" THEN STOP
 840 INPUT  PAPER 5;"Please talk to me
"' PAPER 7;"(No punctuation or capital
s     except I)"''i$: IF i$="" THEN PR
INT  BRIGHT 1'"Don't be stupid!": PAUS
E 50: GOTO 840
 850 PRINT  BRIGHT 1''(CHR$ (CODE i$(1
)-32) AND i$(1)>="a")+(CHR$ CODE i$(1)
 AND i$(1)<"a");i$(2 TO )''
 860 FOR f=1 TO LEN i$: IF i$(f)=" " T
HEN LET i$(f)=" STOP "
 861 NEXT f: LET b$(2 TO )=i$: LET n=L
EN i$+1
 862 RESTORE : FOR f=1 TO 100: READ k$
,x$,y$: LET a$=k$+" STOP ": POKE 65263
,0: POKE 65264,0: LET p=USR 65265: IF
p>0 AND p<n THEN PRINT  PAPER 6;(x$ AN
D r1>.5)+(y$ AND r1<=.5)': GOTO 830
```

```
 880 NEXT f
 882 PRINT  PAPER 6;("I see" AND r=1)+
("Tell me more" AND r=2)+("This is get
ting interesting" AND r=3)+("I'm not s
ure I understand" AND r=0): IF r<4 THE
N LET r=r+1: GOTO 830
 885 PRINT  PAPER 6;"This is boring! P
lease change  the subject": LET r=0:
GOTO 830
 900 DATA "said","Be careful of what o
thers say","Who cares!","sometimes","O
nly sometimes?","Not often?","occasion
ally","Try more often","Only occasiona
lly?","like","I agree","In what way?",
"same","I agree","How?","alike","Exact
ly alike?","Really?","every
body","Surely not!","Really everyone?"
,"everyone","I don't believe that","Re
ally?"
9999 LOAD "FASTCODE"CODE : RUN
```

PROGRAM 6.

```
 800 INK 0: BORDER 7: PAPER 0: CLS : F
OR f=1 TO 170: NEXT f: PAPER 7: CLEAR
: LET r=0: DIM a$(31): DIM b$(64): LET
 k$="": PRINT #0;"@ 1982 Sinclair Rese
arch Ltd": PAUSE 100
 810 CLS: PRINT; PAPER 6; INK 1;".....
.........................X...X.XXXXX
.X.....X......XXXX..XX...X.X.....X....
.X.....X....X.XX...X.X.....X.....X....
.X....X.XXXXXX.XXX...X.....X.....X....
X.XX...X.X.....X.....X.....X....X.XX..
.X.X.....X.....X.....X....X..X...X.XXX
XX.XXXXX.XXXXX..XXXX..X.............
................."
 820 PRINT TAB 3;"This is the Spirit o
f your"' FLASH 1;"       SPECTRUM COM
PUTER       "' FLASH 0;TAB 12;"speakin
g": PAUSE 100: PRINT '"Anyone who play
s such a patheticgame must be in need
of help."'"Let me be of service.": PA
USE 100
 880 NEXT f: IF RND<.2 THEN GOSUB 895:
 REM This is NEW!
 895 PRINT "Excuse me. I need a nervou
s    breakdown.": PAUSE 150: FOR f=1 T
0 20: PAPER 0: CLS: BORDER 7: PAPER 7:
 CLS: BORDER 0: NEXT f: BORDER 7: PAUS
E 50: PRINT "That's better. Now where
was I? Oh yes ..."'': PAUSE 100:RETURN
```

NOTE:- The @ in line 800 is the copyright sign. For each "." in line 810 you should type a space, for each "X" type Graphic Mode/Caps Shift 8. This should spell out "HELLO!".

TABLE A.

213	68	221	42	75	92	17	12
0	221	25	42	75	92	17	44
0	25	94	35	86	235	237	91
88	255	167	237	82	68	77	42
75	92	25	17	49	0	25	221
86	0	122	237	177	234	141	255
237	67	88	255	201	221	229	229
221	35	221	86	0	122	254	226
32	22	42	75	92	17	49	0
25	235	225	221	225	167	237	82
34	88	255	237	75	88	255	201
190	32	3	35	24	218	225	221
225	24	196	33	39	128	1	129
94	17	5	128	237	176	201	

- - - - - - - -

WRITING FOR FORMAT

We are always on the look-out for articles and programs to publish in FORMAT.

Articles can be on any subject related to the Spectrum, DISCiPLE, PLUS D or computing in general. From half a page to a long series.

Don't worry too much about spelling and things like that (the Editor can't spell either) we will sort things out. Just put it down as clearly as you can. It is best if you send your article as a word processor file, on disc or tape, but please include a printed copy so we can look at it straight away.

If you want to include any pictures or diagrams then draw them in black. Send them flat, DO NOT CREASE.

When sending in a program make sure you give clear instructions. Remember to say what equipment is needed to run the program, i.e. memory required, does it need a disc interface, joysticks, ect. If you can, give examples of program screens.

Send your work to our normal address, or give us a ring to talk about it.

SMALL ADS. SMALL ADS.

OVER 1300 Multiface Compatible Pokes on 25+ A4 sheets. Send only £1.30 and 2x1st class stamps to Neil Kurz, 8 Mayfield Close, Old Harlow, Essex, CM17 0LH.

WANTED Spectrum Teletext Adaptor. I saw one advertized about 18 months ago but cant find it now. Ring Gerry Giles on 01 890 9733 after 6pm or at weekends.

TAPESNAP is now available for the 48k and 128k Spectrum. Transfer snapshot files easily to tape. Just select the file number of the snapshot and start the tape.
The programs are supplied on tape and they save themselves to disc. DISCiPLE and PLUS D compatible.
TAPESNAP 48 and TAPESNAP 128 cost £4 each or £6 for both. Please send postal orders or cheques (payable to S.Young) to Shimon Young, 21 Colchester Road, Southend-on-Sea, Essex, SS2 6HW.

YOUR ADVERT

Buying, Selling, Pen Friends, Clubs, etc.

This space is reserved for you. Any PRIVATE advert, up to 30 words (subject to acceptance), will be printed FREE in the next available issue. Any software sold must be original copies, with full instructions and in working order. The publishers will not be held, in any way, responsible for adverts in this column. Trade advertisers should contact the publisher for rates.

* - * - * - *

BACK ISSUES

For members who have missed past issues of FORMAT (or perhaps worn theirs out through constant use) we run a back-issue service. The cost is £1 per issue (£1.25 overseas) incl p&p. Your copies will be sent out with your next monthly issue of FORMAT (provided we receive your order at least a week before). Make cheques (drawn on UK bank or Euro-Cheques, P.O., cash) payable to FORMAT.

AVAILABLE ISSUES

Vol 1 Issues #9 (Apr 1988) - to - #12 (Jul 1988).
Vol 2 Issues #1 (Aug 1988) - to - #7 (Feb 1989).

Note:- Issues 9 to 12 of Vol 1 are now in short supply.

Please WRITE YOUR ORDER ON A SEPARATE PIECE OF PAPER. DO NOT include letters with your order as this will cause delays.

Remember to quote your membership number or we wont be able to send you your order.

· NAMES & NUMBERS ALL YOUR DISCS ·

EASY TO USE

THE ORIGINAL

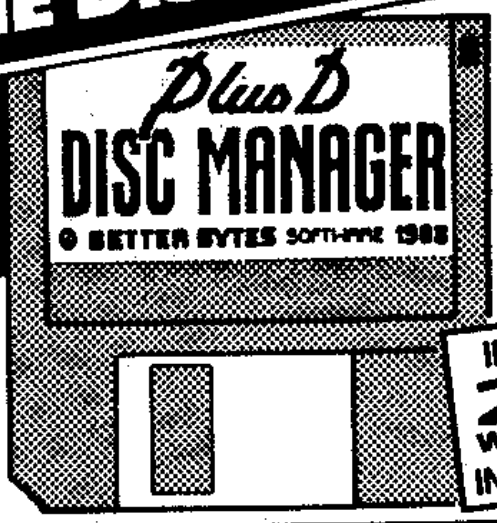
Disciple DISC MANAGER

DISC MANAGER

FOR THE DISCIPLE & PLUS D

PERSONALISE YOUR DISCS · AUTOMATIC CATALOGUER · FAST SEARCH & LOAD · PULL DOWN MENUS ·

A MUST FOR ALL DISCIPLE & PLUS D USERS



NEW UPGRADE VERSION

INCLUDES THE BEST EVER 2 WAY SEARCH IN M/CODE FOR SPECTRUMS

GET TOTAL CONTROL OF YOUR DISC COLLECTION

CUSTOMERS THROUGHOUT THE WORLD – HERE'S WHAT SOME OF THEM SAY...

'IMPRESSIVE' A.S., NEW ZEALAND

'MARVELOUS GRAPHICS' I.A.R., RENFREWSHIRE

'A DELIGHT TO USE' F.W.J., CHIPPENHAM

'VERY USEFUL' R.C.M., NETHERLANDS

'EXCELLENT' R.H. MIDDLESEX

'ESSENTIAL' R.W. CALIFORNIA

The NEW upgrade version of the original DISC MANAGER is the most powerful programme ever written for the DISCIPLE/PLUS D.

Designed to take advantage of Disc Drive ownership, the Manager keeps track of all the programs on all your discs, offering unrivalled benefits and features.

- Storage of up to 27,000 records on one Disc, or 79,920 total. Random File Access.
- Name & Number Discs with fast Autonumber and user pre-defined titles features.
- No typing in of Data. Press a key and Discs are automatically added to appropriate catalogue.
- Fastest ever M/Code Search. 2 modes – Search and Load or Search and List all occurances, then select program to load.
- Special program to print the contents of all your discs with fast sort option. Can selectively print by disc type or number.
- List contents of any disc with ease.
- Multiple erase or rename options.
- Plus many other unique features.
- Comes with 16 page manual and demonstration catalogues.
- Operates with 48K or 128K Spectrums.

Send for the DISC MANAGER today... and you'll soon wonder how you ever Managed without it!

NORMAL PRICE £14.95

SPECIAL INDUG PRICE ONLY

£12.95 INC. P&P

OVERSEAS ORDERS PLEASE ADD £1.50

UPGRADES Existing users can upgrade to the NEW Version.
Send old Manager Disc plus £5.00

STOP PRESS NEW NOW AVAILABLE **DISC ORGANISER**

Recover Erased Programs · Re-organise Directory · Repair Sectors · Wild Card Copy, Erase or Re-name · Supplied on Cassette ONLY **£5**

PROBABLY THE LARGEST PROGRAM EVER WRITTEN FOR THE SPECTRUM! COMPRISES

12 PROGRAMS

6 MAIN – 6 SUPPLEMENTARY

A total of 36 files supplied on disc – over 250K of Program Magic!

· SPECIAL PRINT-OUT PROGRAM · 16 PAGE MANUAL · FLEXIBLE · GREAT GRAPHICS

BETTERBYTES

10 SPITTAL TERRACE · GOSFORTH
NEWCASTLE UPON TYNE NE3 1UT · TEL: (091) 285 6185

SUPPLIED ON 3·5 OR 5·25 DISC · WRITTEN SPECIALLY FOR DISCIPLE/PLUS D

PLEASE STATE YOUR DRIVE TYPE, SIZE, ETC, INDUG MEMBERSHIP No. AND IF REQUIRED FOR DISCIPLE OR PLUS D